EE239 Winter 2015

Project 1

He Ma SID: 904434330 Jiaqi Gu SID: 904235873

Problem 1: Method 1

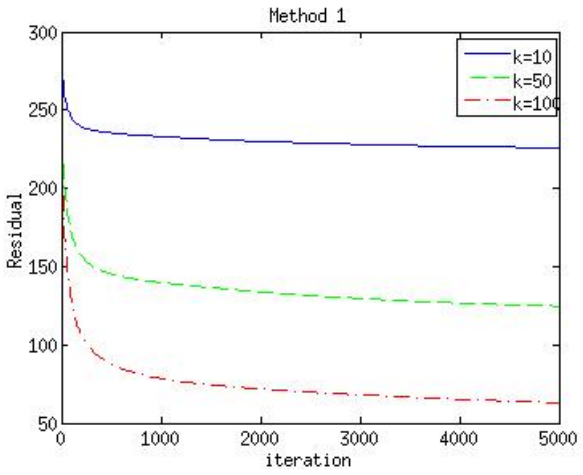
We modified the *wnmfrule* function so that we can access information about the residual:

$$\sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} w_{ij}(r_{ij} - (UV)_{ij})^2}$$

for each iteration.

We also set the default number of iterations to be 5000.

Below is the the plot for how the residual converges when k = 10, 50, 100.



The model converages faster for smaller k. However, the training error is smaller when k is bigger.

After running 5000 iterations, the least square error:

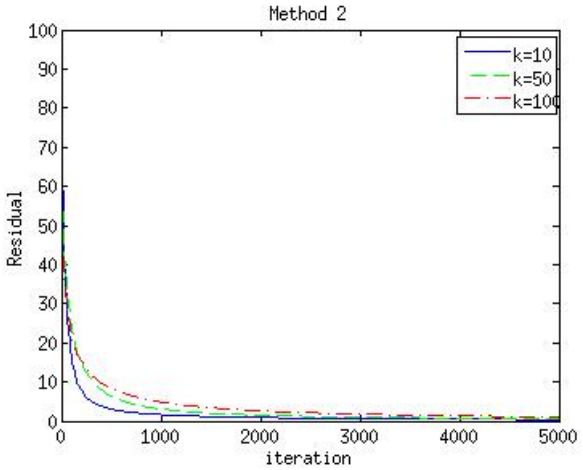
$$\sum_{i=1}^{m}\sum_{j=1}^{n} w_{ij}(r_{ij} - (UV)_{ij})^2$$

is 51014 for k = 10, 15550 for k = 50, and 3982.5 for k = 100.

Problem 2: Method 2

For this question, we switched the values of R and W and then ran the same set of functions as in problem 1.

Below is the the plot for how the residual converges when k = 10, 50, 100.



After running 5000 iterations, the least square error:

$$\sum_{i=1}^{m}\sum_{j=1}^{n} w_{ij}(r_{ij} - (UV)_{ij})^2$$

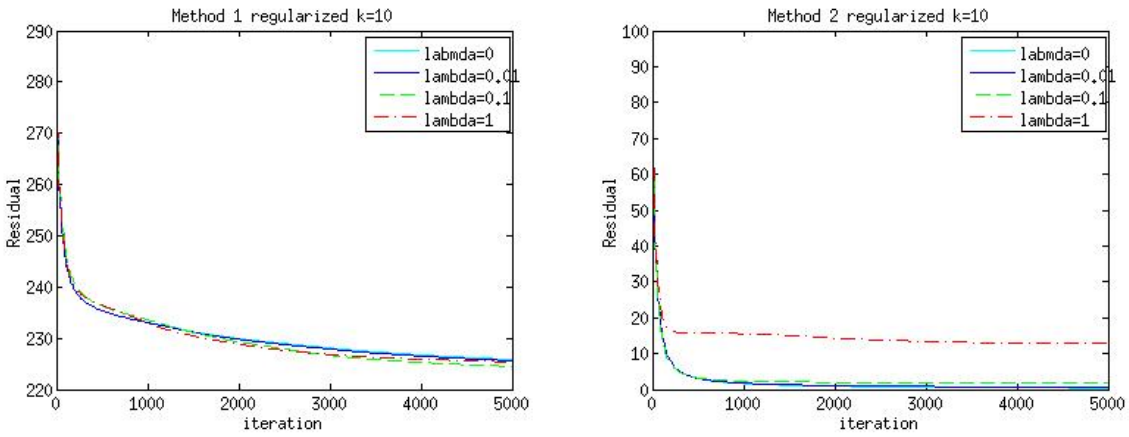
is 0.0413 for k = 10, 0.1086 for k = 50, and 0.3342 for k = 100.

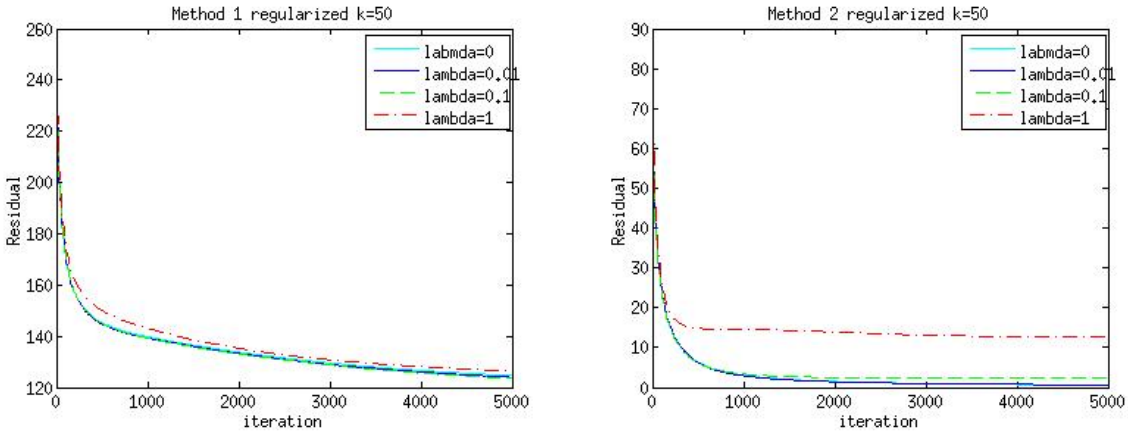
In this case, the training error is smaller when k is smaller. We believe the reason for that is the model converges faster when k is smaller. If we run it using more iterations, the model should converge to a lower value for a larger k.

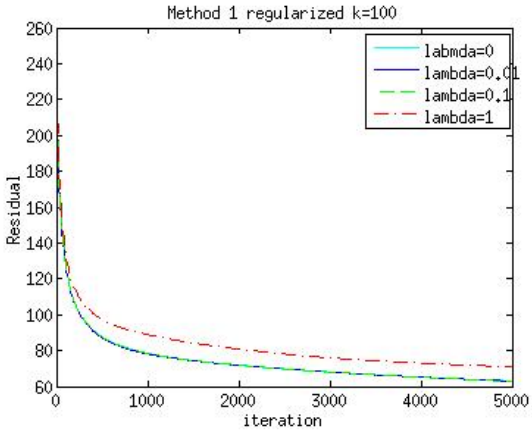
Problem 5
As mentioned in the TA's email, method 2 only works properly when the regularization factor is present. Otherwise, all entrys will converges to 1. So we worked on problem 5 first.

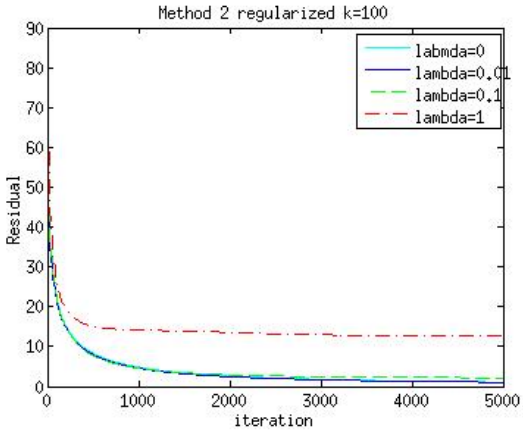
We used the result in the paper *Nonnegative matrix factorization for spectral data analysis* by V. Paul Pauca, J. Piper and Robert J. Plemmons and we implemented the function as $wnmfrule\_sparse$.

Below is the the plot for the residuals when k = 10, 50, 100 and λ = 0.01, 0.1, 1.







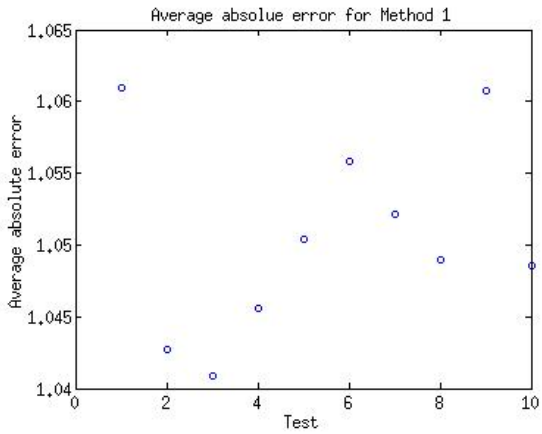


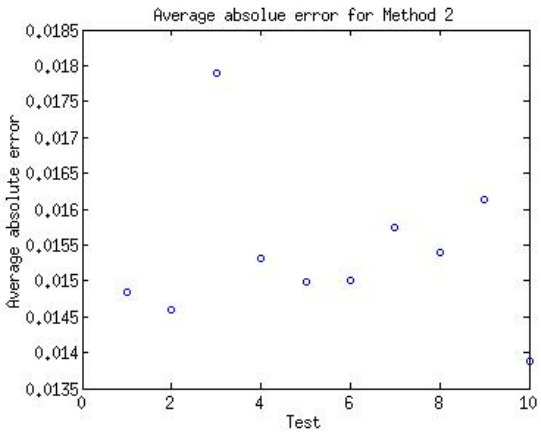
There isn't much difference for training error when $\lambda = 0$, 0.01 and 0.1. However, when λ is 1, training error increases.

For question 3, 4 and 6, we compared the results of method 1 and method 2 when k=100 and $\lambda = 0.1$.

Problem 3

Below is the average absolute error for each testing fold:

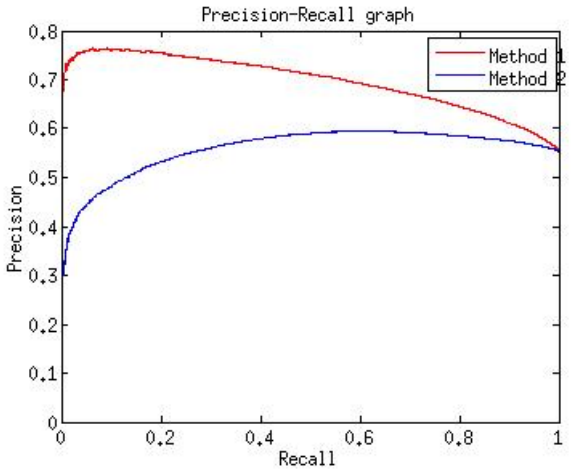




Method 1: k=100, $\lambda = 0.1$:
The average absolute error is 1.0507.
The lowest absolute error is 1.0408.
The highest absolute error is 1.0609.

Method 2: k=100, $\lambda = 0.1$:
The average absolute error is 0.0154.
The lowest absolute error is 0.0139.
The highest absolute error is 0.0179.

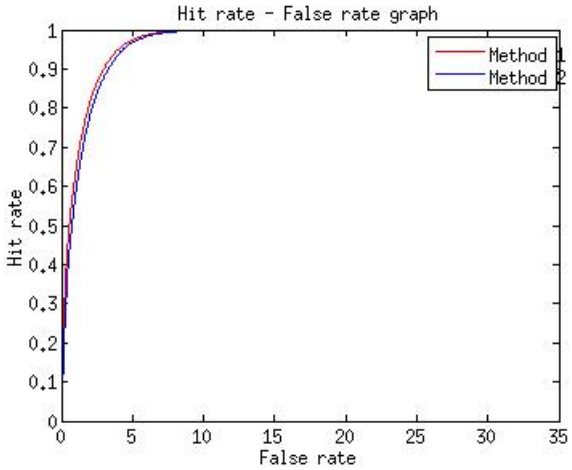
Problem 4
Below is the precision and recall plot for both methods(k=100, λ=0.1):



As shown from the plot, method 1 works better than method 2.

Problem 6
Method 1: k=100, $\lambda = 0.1$, L=5:
Precision: 0.7048
Recall: 0.06
Method 2: k=100, $\lambda = 0.1$, L=5:
Precision: 0.5599
Recall: 0.0477

Below is the Hit rate and False alarm rate plot for both methods(k=100, λ=0.1, L=1 to 1682):



As shown from the plot, method 1 works slightly better than method 2.

How to run the code:
main.m is the file containing the main rountine. It takes about 6 hours to run the code. You can access the work space image from this link.